

REMARKS

Claim Amendments

Claims 1, 6, 9-13, 17-21, 23, 24 and 28-34 are pending, including independent claims 1, 13, 21, 24, 33 and 34.

Applicants have cancelled claim 34; thus rendering moot the rejection under 35 U.S.C. §101.

Applicants have amended independent claim 1 to recite the step of calling an application defined callback that supplies application-specific handling which re-executes the instruction and continues at a next line of code of the application. In view of this amendment, the first step also has been amended to clarify that a thread is executing an instruction of a C++ based application. Support for these amendments is found in the application as originally filed, e.g., page 14, paragraph 0057. No new matter has been added into the application by these amendments.

Similar amendments have been made to independent claims 13, 21, 24 and 33.

Turning to the art rejections, and considering first the rejection of claims 1, 6, 13, 23, 24, 28, 33 and 34 under 35 U.S.C. §103(a), as being unpatentable over Kannan et al. (US Patent 5,818,702, hereinafter called "Kannan") in view of Bak et al. (US Patent 6,415,381 B1, hereinafter called after "Bak") further in view of Anschuetz et al (US Patent 5,305,455), applicants respectfully request reconsideration of the rejection based on the amended claims and for the reasons set out below.

Claim 1

Amended claim 1 recites a method for recovering an application from a runtime fault.

The method receives an exception caused due to a runtime fault in a thread executing an instruction of a C++ based application, and traps the exception before the exception reaches a

top level exception handler. The method translates the trapped exception into a C++ exception that the C++ application is capable of handling. The trapped exception is translated into the C++ exception which is able to be resolved by an application defined C++ exception handler. The method determines if there is an application based C++ exception handler which is capable of resolving the translated exception, terminates the thread that caused the exception when there is no C++ based exception handler which is capable of resolving the translated exception, calling an application defined callback that supplies application-specific handling which re-executes the instruction and continues at a next line of code of the application; and continues execution of the application.

In contrast, none of Kannan, Bak or Anschuetz teach or suggest such a method for recovering an application from a runtime fault by calling an application defined callback that supplies application-specific handling which re-executes instruction and continues at a next line of code of the application.

The Examiner has alleged that Kannan and Bak teach the application is able to handle exceptions by itself programmatically.

Kannan states "In addition to the applications 105, there is installed in the memory 103 a software product 102... comprising a crash guard process 107, an exception handler 115, and a safe message loop 109" (column 4, lines 15-19). Thus, the software product 102 is not part of the applications 105, as also clearly shown in Figure 1. The safe message loop 109 of the software product 102 "replaces, or substitutes for this main event or message loop of any application 105 once the application generates a fatal exception" (column 4, lines 58-61). Accordingly, the applications 105 shown in Kannan do not handle exceptions by themselves.

Bak discloses a system that implements an execution stack, which for each thread "stores frames for each of the functions that have not completed execution" (column 6, lines

52-54). As shown in Fig. 4, the Java runtime system 201, which is not the application (column 6, lines 37-38), has an execution engine 217. The execution engine 217 may use support code 221, which provide functionality relating to exceptions (column 6, lines 48-49). When an exception is generated, the Java run time system searches for an exception handler for the exception within the function and then propagates through the functions on the execution stack (column 11, lines 25-29). Thus, according to Bak, the system that implements the execution stack handles exceptions.

Neither Kannan nor Bak teach or suggest termination of a thread executing an application or calling an application defined callback that supplies application-specific handling which re-executes instruction and continues at a next line of code of the application.

Anshuetz uses a system exception handler 70 to terminate a thread that is executing a process in the operation system and that caused an operating system level exception (column 1, lines 22-34; column 5, lines 9-16). The system exception handler 70 is part of Kernel address space in memory 16 as shown in Figure 2. Anshuetz does not teach or suggest termination of a thread executing an application.

Also, Anshuetz does not teach or suggest calling an application defined callback that supplies application-specific handling which re-executes instruction and continues at a next line of code of the application.

Since none of Kannan, Bak and Anshuetz teach or suggest an application handling exceptions by itself, it is impermissible hindsight to allege it obvious to apply Anshuetz's termination of a thread that is executing a process in the operation system to a thread executing an application. In fact, Kannan is directed to handle a fatal exception generated by the application, and yet does not teach or suggest terminating a thread executing the application. In stead, Kannan proposes to replace an offensive message with a pre-stored message loop. Bak

discusses propagation of exceptions through an execution stack in details, but does not teach how exceptions are handled by a function. These facts clearly show that it is unobvious to apply Anschuetz's operation system level handling to an application level handling.

Furthermore, none of Kannan, Bak and Anschuetz teach or suggest recovering an application from a runtime fault by calling an application defined callback that supplies application-specific handling which re-executes instruction and continues at a next line of code of the application. Accordingly, even if one skilled in the art attempts to combine Kannan, Bak and Anschuetz, he would still fail to achieve the invention as claimed in amended claim 1 of the present application. He would provide an execution stack and propagate exceptions through the functions on the execution stack, replaces the offending message with a pre-stored safe message loop, and terminates a thread executing a process of the operating system. He would not be able to call an application defined callback that supplies application-specific handling which re-executes instruction and continues at a next line of code of the application.

Therefore, it is respectfully submitted that the invention as claimed in amended claim 1 is unobvious and has been patentably distinguished over the combination of Kannan, Bak and Anschuetz.

Independent claims 13, 21, 24, 33, and 34

Independent claims 13, 21, 24 and 33 also have been amended to recite the steps or corresponding elements as recited in amended claim 1. Thus, the above arguments also apply to these claims.

Accordingly, Applicants respectfully submit that claims 13, 21, 24 and 33 also are unobvious and have been patentably distinguished over the combination of Kannan, Bak and Anschuetz.

Dependent claims 6, 23 and 28

Claims 6, 23 and 28 depend on claims 1, 21 and 24, respectively, and are patentable over Kannan and Bak and Anshuetz for the same reasons above addressed relative to claims 1, 21 and 24 (35 U.S.C. 112) as well as for their additional limitations.

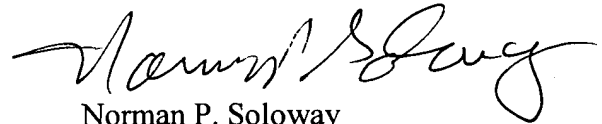
Turning to the rejection of claims 9-12, 17-20 and 29 under U.S.C. 103(a) as being over Kannan in view of Bak further in view of Anshuetz further in view of LeVine et al. (US Patent 6,591,379 B1, hereinafter called "LeVine"), these claims directly or indirectly depend on amended claims 1, 13 and 24. The deficiencies of Kannan, Bak and Anshuetz vis-à-vis claims 1, 13 and 24 are discussed above. Levine does not supply the missing teachings. That is to say LeVine does not teach or suggest any application defined callback that supplies application-specific handling which re-executes instruction and continues at a next line of code of the application. Accordingly, it is respectfully submitted that claims 9-12, 17-20 and 29 also are patentably distinguished over the combination of Kannan, Bak and Anshuetz further in view of LeVine.

Finally, turning to the rejection of claims 30-32 under 35 U.S.C. §103 (a), as being claims are unpatentable over Kannan in view of Bak further in view of Anshuetz further in view of Levine and further in view of Lillevold (US Patent 6,230,284 B1, hereinafter called "Lillevold"), all of these claims directly or indirectly depend on amended claim 24. The differences of the combination of Kannan, Bak, Anshuetz and Levine vis-à-vis claim 24 are discussed above. Lillevold does not supply the missing teachings. That is to say, Lillevold does not teach or suggest any application defined callback that supplies application-specific handling which re-executes instruction and continues at a next line of code of the application. Accordingly, it is respectfully submitted that claims 30-32 also are patentably distinguished over the combination of Kannan, Bak and Anshuetz further in view of LeVine and Lillevold.

Having dealt with all the objections raised by the Examiner, the application is believed to be in order for allowance. Early and favourable reconsideration of the application is respectfully requested.

In the event there are any fee deficiencies or additional fees are payable, please charge them (or credit any overpayment) to our Deposit Account Number 08-1391.

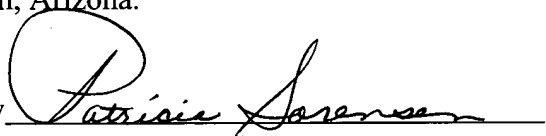
Respectfully submitted,



Norman P. Soloway
Attorney for Applicant
Reg. No. 24,315

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to: MAIL STOP AMENDMENT, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on August 16, 2006, at Tucson, Arizona.

By 

NPS:ps

WAYES SOLOWAY P.C.
30 W. CUSHING STREET
TUCSON, AZ 85701
TEL. 520.882.7623
FAX. 520.882.7643

175 CANAL STREET
MANCHESTER, NH 03101
TEL. 603.668.1400
FAX. 603.668.8567